Reducing-Over-Time Tree for Event-based Data

Reducing-Over-Time Tree for Event-based Data

Shane Harrigan, Sonya Coleman,
Dermot Kerr, Pratheepan Yogarajah
School of Computing, Engineering
and Intelligent Systems
Ulster University
Northern Ireland, UK
BT480QR
Email: {harrigan-s,sa.coleman,d.kerr,p.yogarajah}@ulster.ac.uk

Zheng Fang, Chengdong Wu
Faculty of Robot Science and Engineering
Northeastern University
Liaoning, China
110169
Email: {fangzheng,wuchengdong}@mail.neu.edu.cn

***Abstract*—This paper presents a novel Reducing-Over-Time (ROT) binary tree structure for event-based vision data and sub-types of the tree structure. A framework is presented using ROT, that takes advantage of the self-balancing and self-pruning nature of the tree structure to extract spatial-temporal information. The ROT framework is paired with an established motion classification technique and performance is evaluated against other state-of-the-art techniques using four datasets. Additionally, the ROT framework as a processing platform is compared with other event-based vision processing platforms in terms of memory usage and is found to be one of the most memory efficient platforms available.**

I. Introduction

The development of event-based vision sensors, inspired by the study of neural systems of the retina, such as the Dynamic Vision Sensor (DVS) [1] has led to the emergence of event-based computer vision. An event is a transmission from an event-based vision sensor which indicates that a change in luminance occurred at a particular time t. Event-based vision sensors, such as the DVS, use the Address-Event Representation (AER) [2]. Event-based vision sensors are an alternative to the frame-based vision sensors widely deployed today. Frame-based vision sensors transmit data synchronously in the form of frames (e.g. 30 frames-per-second). Frames are most often 2-D representations of the pixel array values at the time of capture. Apart from the biological-inspiration of event-based vision devices, the main divergence between frame-based and event-based devices is the synchronous and asynchronous rates of data transmission. The synchronized transmission rate of data within frame-based vision devices allows for the capture of static scene details but contributes latency. As demands for faster means of obtaining image data have arising, particular in the research of high-speed robotics, the latency of frame-based devices has begun to take its toll. This issue, particularly in high-speed demanding scenarios, has contributed to the rise of asynchronous devices such as event-based sensors which transmit at event resolution times (e.g. 15 microseconds) [3]. An event-based vision sensor emulates retinal ganglion cell signal behaviour by producing an asynchronous spike, or *event*, [1] when a change in luminance continuously detected by an event-based vision pixel goes beyond an adaptive threshold maintained at the hardware level. Event-based vision sensors offer a high-temporal resolution at a lower power consumption rate compared to traditional frame-based sensors which emit pixel levels cyclically at defined intervals [4]. Each event $e_i \in \langle t_i, l_i, p_i \rangle$ where $i = (0, \ldots, last)$, t is a microsecond timestamp indicating when the change of luminance was detected, $l = \langle x_i, y_i \rangle$, where x, y correspond to pixel coordinates within a 2-D pixel array and p is a binary set $\{1, -1\}$ indicating whether the change of luminance was positive (increasing) or negative (decreasing). The processing of event-based vision data is a new domain within the computer vision field and many of the established techniques, algorithms and tools readily available within conventional frame-based image processing are not compatible with the new event based representation of image data. This paper presents a novel means of processing event-based vision data using a Reduction-Over-Time binary tree structure which is self-balancing and self-pruning. The approach is inspired by the red-black tree structure [5] which can be leveraged to extract edges and corners from event-based vision data. The event-based vision sensor datatype requires a paradigm shift [6] and as such new techniques, algorithms and tools need to be designed for event-based vision data. Current research in this area has primarily focused on adaption of event-based vision data to work with existing frame-based image processing techniques and algorithms [7], [8], [9] negating many of the benefits offered by these sensing devices. In particular, recent publications within event-based vision processing have emphasised the use of machine learning [10], [11], [12], [13], [14], [15], [16], [17] approaches, clustering [18] or using time/active-event surfaces [19] to effectively process these data. Binary trees have been used in traditional image processing for many different tasks [20], [21], [22] but, we are unaware of binary tree structures being used for event-based data; therefore this paper presents a novel binary tree structure which is self-balancing and self-pruning in contrast with the current linear models used for event-data handling/storage [6]. The presented structure offers a novel way of efficiently handling event-based vision data without manipulating or compressing the data as used in other approaches [23].

II. Reduction-Over-Time (ROT) Tree

The Reduction-Over-Time (ROT) tree is a novel self-balancing binary tree structure. The approach is inspired by the red-black tree structure [5] and designed to manage event data and remove nodes corresponding to events which exceed a time limit τ. Each node within the tree contains the data of an event. There are two sub-types of ROT trees: ROT-spatial and ROT-temporal trees. ROT-spatial nodes are indexed by spatial information l and ROT-temporal nodes are indexed by the timestamp t (as discussed in Section I). The self-balancing nature of the ROT tree ensures minimal latency for any subsequent operations interacting with the tree structure. Let a stream of events be $S = \{e_0, e_1, \ldots, e_{last}\}$. Accessing nodes contained within an ROT tree can be expressed as $ROT : Key \longrightarrow n(v_i)|v_i = t_{Key} \cup \{\emptyset\}$ where $Key = \{v_0, \ldots, v_{last}|v \in S(v)\}$, $S(v)$ returns all of the indexing variables within S, node n contains the data of an event e accessed by its t value as well as the references to its parent or children and v is the indexing variable of the ROT tree structure. In ROT-spatial the index variable is l and in ROT-temporal the index variable is t. To perform the pruning operation three steps must occur, firstly all of the nodes n with index n_i less than $t_{current}$ such that $\check{V} = FORGET(V)$ where $FORGET$ is a power function [24], [25] (1) as illustrated in Fig. 1 and $V = t_{current} - ROT(i)$ must be determined (2) with $t_{current}$ being the timestamp of the most recent event added to the tree.

$$FORGET(t) \longrightarrow \frac{0.184}{\log_{10} t^{1.25} + 0.184} \tag{1}$$

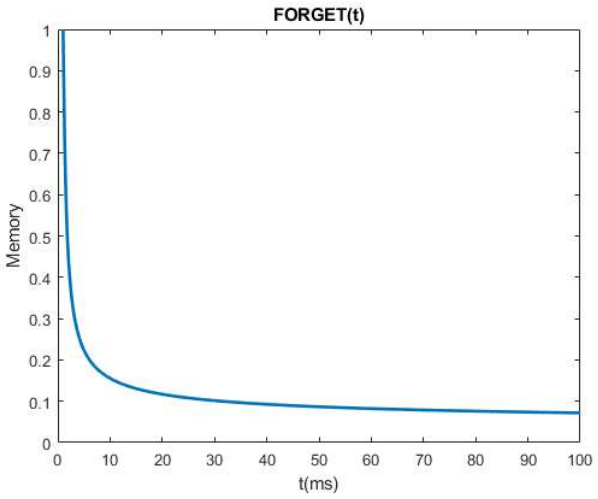

Fig. 1. An illustration of the $FORGET$ function in 1 over t ranging from 0 to 100.

$$nodes : ROT, V \longrightarrow \{ROT(i)|\check{V} > \tau\}, \tag{2}$$

where $nodes$ returns the index i of nodes n within a binary tree structure ROT which exceed threshold τ. Secondly, the largest value $node_{largest}$ of $nodes$ must be determined using the function max which returns the highest value within a collection such that:

$$node_{largest} \longrightarrow max(nodes). \tag{3}$$

Thirdly, all nodes up to $node_{largest}$ must be removed from the tree leaving only a balanced tree of nodes that are within a time limit such that:

$$prune : ROT, node_{largest} \longrightarrow \{ROT(i)|ROT(i) > node_{largest}\}, \tag{4}$$

where $prune$ returns a new binary tree structure containing events which exceed the threshold τ, $ROT(i)$ is a node within the binary tree structure ROT at i and $node_{largest}$ (3) is the maximum value removed from the previous tree. The pruning operation focuses on the discovery of the right most node $ROT(i)$ which violates a declared threshold T, such that $V > T$. If the node is a parent, any left children are also removed from the tree. If the node is the right child of a parent node then the parent can also be removed. The searching stage of the operation can be performed recursively from the root node while being right-leaning (Algorithm 1) for performance making the search logarithmic ($\mathcal{O}(\log n)$) in time complexity. The inherent self-balancing architecture of the ROT binary tree structure also makes deletion logarithmic in time complexity ($\mathcal{O}(\log n)$), but this can be reduced to a constant time complexity ($\mathcal{O}(1)$) if the tree is treated as a list or as a result of localised use of the DSW algorithm [26]. Additionally, the insertion of the values into a ROT-temporal tree is $\mathcal{O}(\log n)$ as, similar to deletion, addition is just a search operation.

Algorithm 1 Function to find index of the maximum violating node recursively where ROT is the binary tree structure being searched, T is the threshold and C is the latest event timestamp value.

Require: $ROT \neq$ **NIL**
function FINDMAXVIOLATION(node, T, C)
  if $node =$ **NIL then**
    return
  end if
  FINDMAXVIOLATION($node_{right}, T, C$)
  if $(C - node_i) > T$ **then**
    return i
  end if
  FINDMAXVIOLATION($node_{left}, T, C$)
end function

III. ROT feature framework

A feature extraction framework using the each sub-type of ROT tree is presented to leverage the tree structure. Two ROT-spatial trees representing positive or negative polarity are formed from a ROT-temporal tree. Each node within a ROT-spatial tree represents a physical l_i pixel co-ordinate. Considering the DVS128 event sensor [1], there can be a

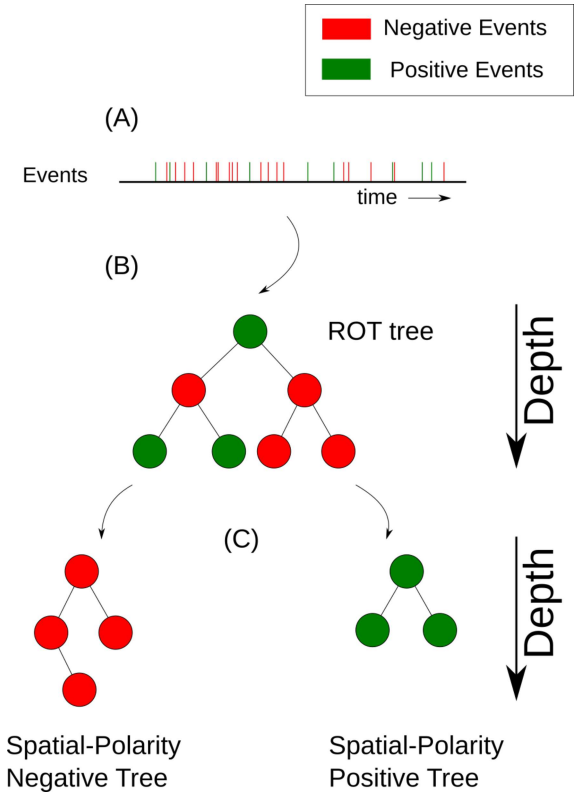

Fig. 2. Illustration of the ROT framework showing an event data stream (A) with a portion of the data shown referenced within a ROT-temporal (B) and the ROT-spatial positive/negative trees (C) generated from the ROT-temporal nodes.

maximum of 16384 nodes within each ROT-spatial tree. Fig. 3(C) shows a re-constructed frame using the ROT-spatial positive tree.

A. *Edge-event Filter*

In frame-based image processing, an edge is often defined as the point of an abrupt discontinuity in image illumination [27]. The data obtained through event-based vision means that the standard definition for edges within frame vision processing no longer applies. Each event produced by event-based vision devices is an indication of a detected luminance change at a spatial-temporal location. The asynchronous temporal aspect of event data means that the frame-based definition of an abrupt discontinuity is no longer valid (demonstrated in Fig. 4 as all events are abrupt discontinuities).

Within the ROT framework we define an edge as being a node within the ROT-temporal tree and also existing in both the ROT-spatial positive and ROT-spatial negative trees and which can be easily determined using Algorithm 2 such that l_i is contained within both trees. In this paper, events containing the unique ROT-spatial property (coming from the same node across the two distinctive trees) are known as edge-events. Fig. 3(D) shows a frame constructed from event data containing event responses from a DVS128 whilst observing a checkerboard pattern moving relative to the device; blue dots indicate that the event was found in the ROT-spatial positive tree at a lower depth compared to its ROT-spatial negative counterpart and red dots indicate that the event was found in the ROT-spatial negative tree at a lower depth compared to its ROT-spatial positive counterpart.

Algorithm 2 Function to perform an AND operation based on the spatial index l of a node A with a ROT-spatial tree B resulting in a ROT-spatial tree O containing nodes which had an event within both trees. The ADDTO function takes an event A_i and adds it to tree O. The CONTAINS function checks if the index i of the current node exists within ROT-spatial binary tree B using the standard search operation.

Require: $O \longrightarrow$ROT-spatial Tree
1: **function** ROTAND(A, B, O)
2: **if** $(A =$ **NIL**$)$ OR $(B =$ **NIL** $)$ **then**
3: **return**
4: **end if**
5: **if** $(A_{left} \neq$ **NIL** $)$ **then**
6: ROTAND(A_{left}, B, O)
7: **end if**
8: **if** $(A_{right} \neq$ **NIL** $)$ **then**
9: ROTAND(A_{right}, B, O)
10: **end if**
11: **if** CONTAINS$(i,\ B)$ **then**
12: ADDTO$(O,\ A_i)$
13: **end if**
14: **end function**

B. *Corner-event Filter*

Similar to the issue of edge definition in Section III-A, the definition of a corner as understood from frame-based image processing does not fit the representation found in event data. In frame-based images a corner can be simplistically explained as the intersection of two edges. Fig. 3(D) in Section III-A contains edge-events where the depth of the node containing the data in both the ROT-spatial positive and negative ROT-spatial trees determines the type of event when constructing a frame. A similar approach may be taken to identify corners where the depth of both nodes from the root of their respective trees only varies by $+1/-1$ then it is likely to be a corner as the ROT-spatial trees represent the spatial information .

IV. Experiments and Results

Two experiments are conducted using the ROT feature framework discussed in Section III. The first experiment in Section IV-A evaluates an existing template matching framework with the ROT framework in terms of matching accuracy. Section IV-B presents a second experiment which compares the ROT framework with other existing frameworks in terms of memory usage. In both experiments τ was set to 0.4 memory (Fig. 1) meaning that a temporal difference producing 0.4 or less using (1) would result in the corresponding node being removed.

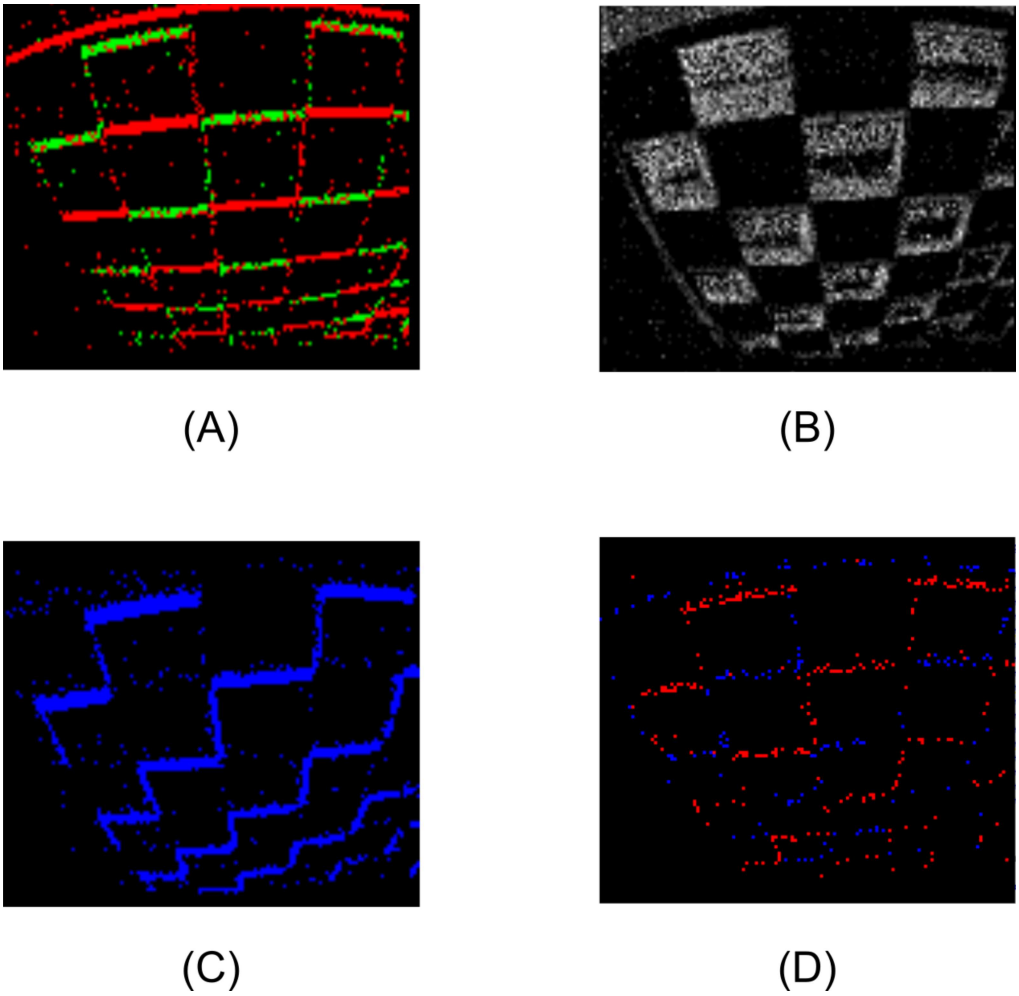

Fig. 3. Screen captures from the ROT framework outputs showing (A) the event data from an event-based vision sensor presented with a checkerboard stimulus (green represents an positive polarity and red represents a negative polarity), (B) the accumulation of the event data over time into a gray-scale image, (C) an image formed from a ROT-spatial positive tree after the edge algorithm (Section III-A) is used and (D) an image formed from ROT-spatial positive and negative trees (using Algorithm 2) with blue indicating positive polarity and red indicating negative polarity; if events exist in both ROT-spatial trees in (D) then the depth of each node is compared with the node closest to the root being displayed over the other.

A. Classification Accuracy Experiment

The ROT framework is used to extract only edge-events (Section III-A) from the stream of events S resulting in a filtered event-based vision data stream E consisting of only edge-events. E was passed as input to a template matching framework for event-data known as P-TED framework based on a event descriptor [28] which converts event data into a motion feature vector (representing the observed motion of a scene) and a pattern feature vector (representing the underlying event response pattern produced as a result of scene change). The pattern feature vector from [28] was replaced with a single ROT-temporal binary tree structure as the tree structure naturally retains events temporally greater than 0.4 in memory (1) and mitigates the need for a sliding window implementation over the event pattern; the ROT implementation for P-TED was referred to as P-TED(ROT). The original P-TED setup was used as a control and is referred to as P-TED in the results. Additionally, the Distribution Aware Retinal Transform (DART) [29] framework was also used for comparison. DART is a state-of-the-art event descriptor formation and framework approach to representing motions in an analyzable manner within the field of event-based computer vision. DART encodes the structural context within event data streams into log-polar grids to simulate the cone photoreceptor distribution within the fovea of of the retina in primates [30]. DART makes use of machine learning and introduces a learning pipeline to train a support vector machine (SVM) to learn from DART descriptors. The P-TED and DART algorithms made use of the background activity filter [31] to reduce noise from the event-data stream while the P-TED(ROT) relied on the edge-event filter presented in Section III-A.

As event-based vision processing is a relatively new field only a small number of datasets are available to evaluate methods. The most prominent event-based datasets are derived from legacy computer vision datasets and therefore suffer from legacy frame-based restrictions [32], [33]. In this paper four of

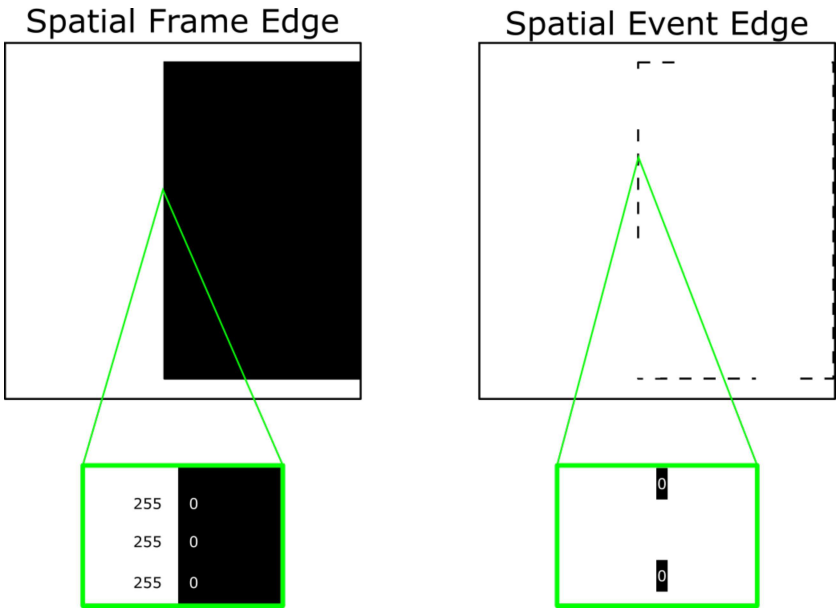

Fig. 4. An illustration of how an edge is represented spatially in frame and event data for comparison. An edge, in frame-based image processing, is understood to be the point of an abrupt discontinuity in image illumination; 255 and 0 are the pixel values for white and black colour respectfully.

the most prominent event-based datasets are used to evaluate P-TED(ROT) and P-TED. **N-MNIST** [34] which consists of 70,000 event responses to the original MNIST [35] data. The database consists of 10 classes representing the digit range $0-9$. This dataset was created using saccade-based movement of each MNIST entry tracing an isosceles triangle on an LCD screen while a asynchronous time-based image sensor (ATIS) [36] produced the event response to the movement for each sample within the database. **MNIST-DVS** [37] is similar to the N-MNIST dataset but contains recordings of 30,000 event responses to the MNIST database samples which consists of 10 classes representing the digit range $0-9$. Each digit within the MNIST database has induced programmatic motion while being displayed on a high contrast screen to minimise event artefacts; these motions are recorded by a DVS128 for $2-3$ seconds each. The MNIST-DVS dataset poses an interesting challenge to object recognition in event-based vision processing since the object is continuously revolving there is no apparent break in the event data. **MNIST-FLASH-DVS** [37] is identical to the MNIST-FLASH-DATASET dataset with the exception that, instead of programmatic motion, the captured event response was produced by the MNIST digit flashing five times from a stationary position. The MNIST-FLASH-DVS dataset is an easier version of the MNIST-DVS for object recognition since the period flashing means that there are ”reset” points in the data stream where machine learning tools and algorithms can forget all previous events during this period of time. **CIFAR10-DVS** [38] consists of 10,000 event responses corresponding to the original CIFAR10 dataset [39]. The dataset consists of 10 classes representing 1,000 event response recordings for airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks images. The dataset used a 128×128 DVS [1] camera though the manufacturer or model was not specified. The CIFAR10-DVS is probably one of the most difficult datasets for event-based vision processing relating to object classification since the objects shown in the dataset for each class are linked by a textual description (i.e. aeroplane) but might appear aesthetically different (i.e. a Boeing 747 looks vastly different to a Airbus A300). DART was trained with a 70:30 split for training and testing. The first entry for each class within each dataset was used as the template for that class within the P-TED framework. Table I presents the accuracy results obtained from using the P-TED, P-TED(ROT) frameworks and the DART learning pipeline on the four datasets discussed above. The P-TED(ROT) outperformed the original P-TED setup across all four of the datasets. This is likely due to the noise minimising affect that edge identification has on event-data as discussed in Section III-A. The result of the CIFAR10-DVS in particular shows the improvement achieved. The CIFAR10-DVS dataset is made up of event responses to various images such as airplanes and cats which possess a lot of noise. The DART learning pipeline poorer performance is likely to do with the sensitivity to noise which could not be filtered out effectively by the background activity filter.

TABLE I
CLASSIFICATION ACCURACY RESULTS OF DIFFERENT EVENT-BASED VISION DATASETS OBTAINED USING THE P-TED TEMPLATE MATCHING FRAMEWORKING USING P-TED AND P-TED(ROT).

DATASET\FRAMEWORK	P-TED (%)	P-TED(ROT) (%)	DART (%)
N-MNIST	93.09	98.44	91.12
MNIST-DVS	91.73	96.71	86.74
MNIST-FLAST-DVS	94.80	96.93	93.20
CIFAR10-DVS	58.55	71.80	51.74

TABLE II
RESULTS OF THE MEMORY USAGE EXPERIMENT OF POPULAR EVENT-BASED VISION PROCESSING PLATFORMS IN COMPARISON TO THE ROT FRAMEWORK IN ORDER OF LEAST TO MOST MEMORY USAGE.

Platform	Memory(Mb)
cAER	56
ROT Framework	62
Tarsier, Sepia, Chameleon (TSC)	81
pyAER	186
python-aer	196
jAER	1262

B. Memory Usage Experiment

This experiment seeks to compare the ROT framework with other event based processing architectures in terms of memory usage. Most popular platforms are based on some variation of list structures such as a Queue, stack, or double-linked list [40]. In this experiment the ROT binary tree structure was used to store event data entirely rather than just referencing them. In the experiment each platform was tasked with reading, filtering and displaying the same event data file; this process was repeated five times and the highest memory consumption result (rounded) was taken as the final value. Each platform used a popular filter known as the background activity filter (see jAER documentation for detailed explanation of this filter) except for the ROT framework which instead used the edge-event filter (Section III-A). The platforms used in the experiment are pyAER[1], python-aer[2], cAER[3], jAER[4], Tarsier, Sepia, Chameleon (TSC)[5] and the ROT Framework.

Table II presents the results obtained from the memory experiment. The cAER platform performed the best in terms of minimal memory usage, likely due to the minimal nature of the platform and the use of efficient and mature libraries for handling imagery etc. The ROT framework came second in terms of minimal memory usage, even though the ROT framework did come with an overhead from the Java Virtual Machine management. This is promising as implementing the framework using a native language could demonstrate further improvements. Memory consumption was minimised because the ROT binary tree structure manages data efficiently while facilitating fast in-memory access. The TSC platform was third for memory usage combined. The ability to use each library separately with a much lower memory cost than the combined memory usage cost means that TSC is a suitable existing platform for neuromorphic devices as it is highly extensible.

The pyAER and python-aer platforms came fourth and fifth respectfully. The overhead of these frameworks is likely to be the python implementation which while versatile, was not constructed with efficiency of memory usage in mind. The jAER platform was the worst performing. jAER is a substantial application compared with the other platforms with a lot of functionality. It is memory intensive but highly extensible. The availability of popular algorithms tend to compensate for its poorer performance. Fig. 6 shows a capture taken from the Netbeans environment profiler of jAER (Fig. 6A) and the ROT (Fig. 6B) framework.

[1]https://github.com/duguyue100/pyaer/
[2]https://github.com/darioml/python-aer
[3]https://github.com/llongi/caer
[4]https://github.com/SensorsINI/jaer
[5]https://github.com/neuromorphic-paris

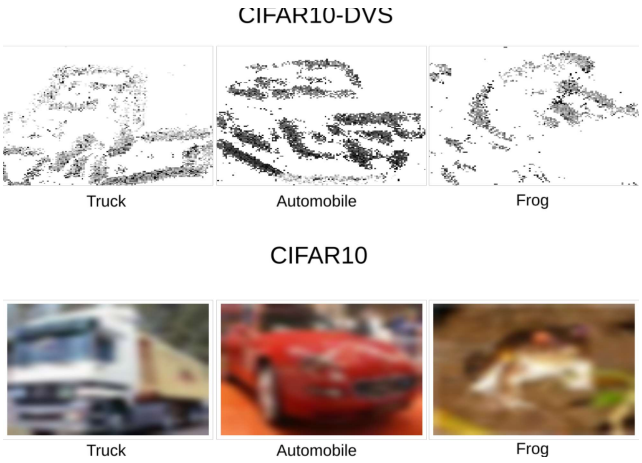

Fig. 5. Samples from the truck, automobile and frog data entries from the CIFAR10 and CIFAR10-DVS datasets.

V. CONCLUSION

This paper presents the Reducing-Over-Time (ROT) binary tree structure for event-based vision processing which uses self-balancing and self-pruning mechanisms (Section II). A framework based around the use of the ROT tree sub-types is also presented (Section III). A definition for an edge within the ROT framework is presented and a filter resulting in edge-events (events with a high probability of being edges) is presented (Section III-A). This framework is coupled with an existing template matching architecture and comparative results demonstrate it outperforms other techniques when using four prominent event based datasets. Additionally, the framework is evaluated against existing event-data processing frameworks in terms of memory performance and ROT demonstrates highly efficient memory usage (Section IV). Future work will include exploring three ROT trees working in cohesion with varying time limits to create scale representations of motion; additionally, corner detection within a tree of edge-events (Section III-A) and different forms ROT tree combination, comparison and pruning will be explored. The identification of corners using the ROT-spatial tree is also viable and will be explored in future work.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] K. A. Boahen, "Communicating neuronal ensembles between neuromorphic chips," in *Neuromorphic systems engineering*. Springer, 1998, pp. 229–259.

[3] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-dof pose tracking for high-speed maneuvers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2761–2768.

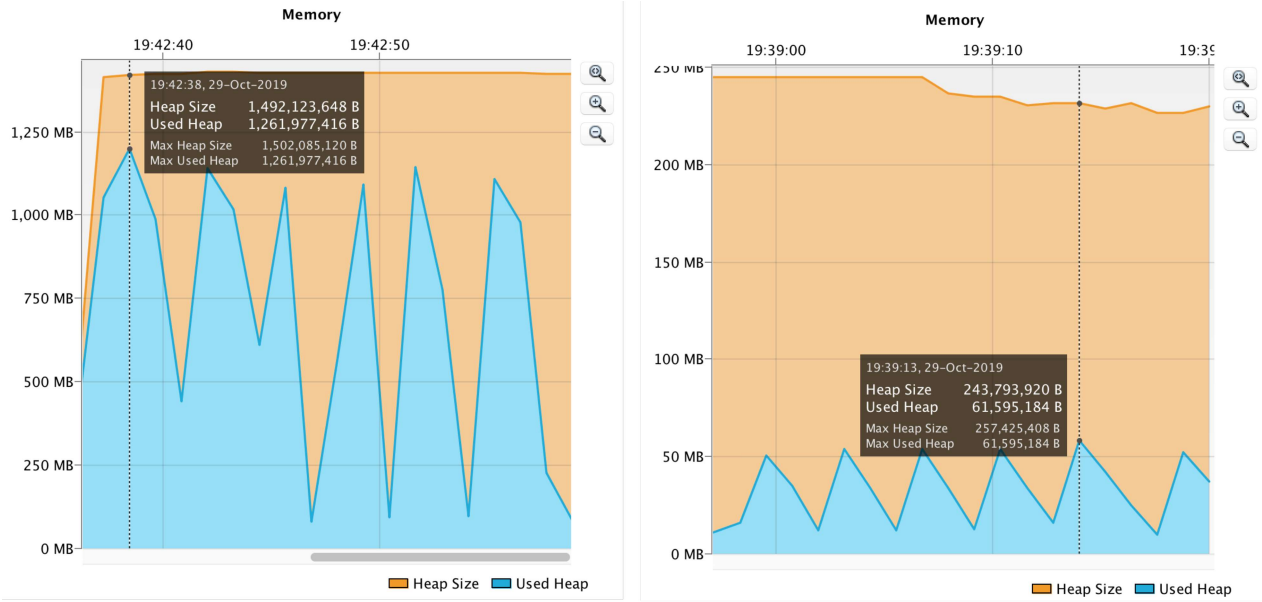

Fig. 6. A capture of the profiler output for memory usage within jAER (A) and ROT framework (B) in Netbeans with the highest memory usage information highlighted.

[4] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[5] L. J. Guibas and R. Sedgewick, "A dichromatic framework for balanced trees," in *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*. IEEE, 1978, pp. 8–21.

[6] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[7] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[8] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4144–4149.

[9] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015.

[10] B. Ramesh, H. Yang, G. M. Orchard, N. A. Le Thi, S. Zhang, and C. Xiang, "Dart: distribution aware retinal transform for event-based cameras," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[11] P. Kirkland, G. Di Caterina, J. Soraghan, Y. Andreopoulos, and G. Matich, "Uav detection: a stdp trained deep convolutional spiking neural network retina-neuromorphic approach," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 724–736.

[12] T. A. Gibson, S. Heath, R. P. Quinn, A. H. Lee, J. T. Arnold, T. S. Sonti, A. Whalley, G. P. Shannon, B. T. Song, J. A. Henderson *et al.*, "Event-based visual data sets for prediction tasks in spiking neural networks," in *International Conference on Artificial Neural Networks*. Springer, 2014, pp. 635–642.

[13] J. C. Thiele, O. Bichler, and A. Dupret, "Event-based, timescale invariant unsupervised online deep learning with stdp," *Frontiers in computational neuroscience*, vol. 12, p. 46, 2018.

[14] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[15] Y. Wang, B. Du, Y. Shen, K. Wu, G. Zhao, J. Sun, and H. Wen, "Ev-gait: Event-based robust gait recognition using dynamic vision sensors," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, "Graph-based object classification for neuromorphic vision sensing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 491–501.

[18] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," *arXiv preprint arXiv:1904.01293*, 2019.

[19] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit, "Speed invariant time surface for learning to detect corner points with event-based cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 245–10 254.

[20] J. A. Robinson, "Efficient general-purpose image compression with binary tree predictive coding," *IEEE Transactions on Image Processing*, vol. 6, no. 4, pp. 601–608, 1997.

[21] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, 2000.

[22] B. Chaudhuri, "Applications of quadtree, octree, and binary tree decomposition techniques to shape analysis and pattern recognition," *IEEE transactions on pattern analysis and machine intelligence*, no. 6, pp. 652–661, 1985.

[23] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," *arXiv preprint arXiv:1904.08245*, 2019.

[24] H. Ebbinghaus, "Memory: A contribution to experimental psychology (ha ruger & ce bussenius, trans.). new york, ny, us," 1913.

[25] J. M. Murre and J. Dros, "Replication and analysis of ebbinghaus' forgetting curve," *PloS one*, vol. 10, no. 7, 2015.

[26] Q. F. Stout and B. L. Warren, "Tree rebalancing in optimal time and space," *Communications of the ACM*, vol. 29, no. 9, pp. 902–908, 1986.

[27] H. G. Barrow and J. M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial intelligence*, vol. 17, no. 1-3, pp. 75–116, 1981.

[28] S. Harrigan, S. Coleman, D. Kerr, P. Yogarajah, Z. Fang, and C. Wu, "Neuromorphic event-based action recognition," 2019.
[29] B. Ramesh, H. Yang, G. Orchard, N. A. L. Thi, and C. Xiang, "Dart: distribution aware retinal transform for event-based cameras," *arXiv preprint arXiv:1710.10800*, 2017.
[30] E. L. Schwartz, "Spatial mapping in the primate sensory projection: analytic structure and relevance to perception," *Biological cybernetics*, vol. 25, no. 4, pp. 181–194, 1977.
[31] H. Liu, C. Brandli, C. Li, S.-C. Liu, and T. Delbruck, "Design of a spatiotemporal correlation filter for event-based sensors," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 722–725.
[32] L. R. Iyer, Y. Chua, and H. Li, "Is neuromorphic mnist neuromorphic? analyzing the discriminative power of neuromorphic datasets in the time domain," *arXiv preprint arXiv:1807.01013*, 2018.
[33] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, 2018.
[34] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.
[35] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[36] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *2008 IEEE International Symposium on Circuits and Systems*. IEEE, 2008, pp. 2130–2133.
[37] T. Serrano-Gotarredona and B. Linares-Barranco, "Poker-dvs and mnist-dvs. their history, how they were made, and other details," *Frontiers in neuroscience*, vol. 9, p. 481, 2015.
[38] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: An event-stream dataset for object classification," *Frontiers in neuroscience*, vol. 11, p. 309, 2017.
[39] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[40] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. John Wiley & Sons, 2014.